

LABIC

Modelos de Neurônios

- Características Básicas
- Modelo de Neurônio
- Estrutura da Rede
- Algoritmo de Aprendizado

REDES NEURAIS - RAFR

1

LABIC

I - PERCEPTRON

$y = f(\sum_i w_i x_i - \theta)$

F é função sinal

REDES NEURAIS - RAFR

2

LABIC

Perceptron

- O algoritmo usado para ajustar os param. Livres desta rede apareceu num proc. De aprendizado desenvolvido por Rosembatt (1958, 1962).
- Ele provou que se os padrões usados para treinar são Linearmente separáveis, então o alg. Converge e a superf. de decisão tem a forma de um hiperplano entre duas classes.

REDES NEURAIS - RAFR

3

LABIC

Perceptron

- É constituído de apenas 01 neuônio e como tal só consegue classificar padrões envolvendo apenas 2 classes, que devem ser linearmente separáveis.
- A regra de decisão é designar x a classe C1 se a saída $y = +1$ e a classe C2 se a saída é -1 .

REDES NEURAIS - RAFR

4

LABIC

Perceptron

- Existem duas regiões separadas pelo hiperplano: $\sum w_i x_i - \Theta = 0$
- Se o espaço for o R^2 , a região. de separação é uma reta

REDES NEURAIS - RAFR

5

LABIC

Teorema de Convergência

- Seja o vetor de entrada:
 $X(n) = [-1, x_1(n), x_2(n), \dots, x_p(n)]^T$

E seu correspondente vetor peso:
 $W(n) = [\Theta(n), w_1(n), w_2(n), \dots, w_p(n)]^T$

A saída pode ser descrita na sua forma compacta:
 $v(n) = w^T(n) \cdot X(n)$

A equação $w^T x = 0$, plotada num espaço p-dimensional define um hiperplano entre duas classes diferentes.

REDES NEURAIS - RAFR

6

LABIC

Teorema de Convergência

- Seja $X1 = \{x^1(1), x^1(2), \dots\}$ ser o conjunto de vetores de treinamento que pertencem a classe C1
- Seja $X2 = \{x^2(1), x^2(2), \dots\}$ ser o conjunto de vetores de treinamento que pertencem a classe C2
- $X = X1 \cup X2$

REDES NEURAIS - RAFR

7

LABIC

Teorema de Convergência

- $X1$ e $X2$ para treinar o classificador \rightarrow ajuste do vetor peso W , de tal forma que as classes C1 e C2 fiquem separadas.
- Então duas classes são ditas linearmente separáveis se existe um vetor peso W .
- Analogamente, se as 2 classes são linearmente separáveis, então \exists um vetor W tal que: $w^T x \geq 0, \forall x, x \in C1$ (X)
- $w^T x < 0, \forall x, x \in C2$

REDES NEURAIS - RAFR

8

LABIC

Teorema de Convergência

- Portanto, dados os subconj. $X1$ e $X2$, o probl. De treinamento do Perceptron consiste em encontrar um vetor W que satisfaz as duas desigualdades (X).
- O alg. para atualização do vetor W pode ser formulado:
- Se $x(n)$ é classif. Corretamente então:
 $w(n+1) = w(n)$ se $w^T x \geq 0, x \in C1$
 $w(n+1) = w(n)$ se $w^T x < 0, x \in C2$

REDES NEURAIS - RAFR

9

LABIC

Algoritmo

- Caso contrário, o vetor peso é atualizado:
 $w(n+1) = w(n) - \eta(n) x(n)$, se $w^T x \geq 0, x \in C2$
- $w(n+1) = w(n) + \eta(n) x(n)$, se $w^T x < 0, x \in C1$
- onde $\eta(n)$ é o param. Veloc. aprendizado
- Se $\eta(n) = \eta$ - param veloc. é fixo

REDES NEURAIS - RAFR

10

LABIC

Convergencia

- A convergencia será provada com $\eta=1$
 $W(0) = 0$; suponha que $w^T(n)x(n) < 0$ para $n=1,2,\dots$ e que um vetor de entrada $x(n) \in$ ao conjunto $X1$, isto é,
 A segunda condição de (X) é verdadeira.
 Então, a correção deve ser realizada de acordo com:
 $w(n+1) = w(n) + x(n)$

REDES NEURAIS - RAFR

11

LABIC

Convergência

- Usando a condição inicial $w(0) = 0$, nós podemos iterativamente resolver esta eq. p/
 $w(n+1) = x(1) + x(2) + \dots + x(n)$ (1)
- Desde que as classes C1 e C2 são assumidas L.S., \exists 1 solução $w_0 / w_0^T x(n) > 0, n=1,2,\dots \in X1$.
- Seja $d = \min_{x(n) \in X1} w_0^T x(n)$

REDES NEURAIS - RAFR

12

Convergência

- Multiplicando ambos os membros de (1) por w_0^T obtemos:

$$w_0^T w(n+1) = w_0^T x(1) + w_0^T x(2) + \dots + w_0^T x(n)$$

$$\geq n d$$

Usando a desigualdade de Cauchy, tem-se:

$$n^2 d^2 \leq [w_0^T w(n+1)]^2 \leq \|w_0\|^2 \|w(n+1)\|^2$$

Ou $\|w(n+1)\|^2 \geq n^2 d^2 / \|w_0\|^2$ (2)

LABIC

REDES NEURAIS - RAFR

13

Convergência

- Considerando a equação (X):

$$\|w(k+1)\|^2 = \|w(k)\|^2 + \|x(k)\|^2 + 2w^T(k) \cdot x(k)$$

$$\leq 0$$

$$\|w(k+1)\|^2 \leq \|w(k)\|^2 + \|x(k)\|^2$$

Ou, equivalente:

$$\|w(k+1)\|^2 - \|w(k)\|^2 \leq \|x(k)\|^2, k=1, 2, \dots, n$$

Adicionando estas desigualdades para $k=1, 2, \dots, n$ e $w(0)=0$,

$$\|w(n+1)\|^2 \leq \sum_k \|x(k)\|^2 \leq n \beta \quad (3)$$

Onde $\beta = \max \|x(k)\|^2$ pertencente ao X_1 .

LABIC

REDES NEURAIS - RAFR

14

Convergência

- A eq. (3) coloca que a norma eucl. ao quadr. do vetor peso $w(n+1)$ cresce linearmente com o no. de iterações.

Isto entra em conflito com a eq. (2).

n não pode ser maior que n_{\max} para os quais as eq. 2 e (3) valem com o sinal de igualdade:

$$n_{\max}^2 d^2 / \|w_0\|^2 = n_{\max} \beta$$

- $n_{\max} = \beta \|w_0\|^2 / d^2$

Portanto, supondo que w_0 existe a regra de atualiz. Deve terminar em n_{\max} iterações

LABIC

REDES NEURAIS - RAFR

15

Modelos de Neurônios

$x_0 = -1$

$w_0 = \theta$ (threshold)

x_1, x_2, \dots, x_n

w_1, w_2, \dots, w_n

Σ

$y = f(\sum_i w_i x_i - \theta)$

F é função sinal

LABIC

REDES NEURAIS - RAFR

16

Modelos de Neurônios

$x_0 = +1$

$w_0 = b$ (bias)

x_1, x_2, \dots, x_n

w_1, w_2, \dots, w_n

Σ

$y = f(\sum_i w_i x_i)$

F é função sinal

LABIC

REDES NEURAIS - RAFR

17

CARACTERISTICAS BASICAS

$$y_j = \text{sgn}(\sum_i x_i w_{ij})$$

- Regra de propagação
- Função de ativação: função sinal
- Topologia: uma única camada de processadores
- Algoritmo de Aprendizado: $\Delta w_{ij} = \eta x_i (t_j - y_j)$ (é do tipo supervisionado)
- Valor de Entrada/Saída: Binários $t = 1$ ou -1

LABIC

REDES NEURAIS - RAFR

18

LABIC Finalidade do Termo *Bias*:

(a) without bias (b) with bias

$\sum_i x_i w_{ij} = 0$ Define um hiperplano passando pela origem

$\sum_i x_i w_{ij} + \theta_i = 0$ Desloca-se o hiperplano da origem

RAFR REDES NEURAIS - RAFR 19

LABIC **ALGORITMO DE APRENDIZADO**
Regra Delta - LMS

- 1) iniciar os pesos sinápticos com valores randomicos e pequenos ou iguais a zero;
- 2) aplicar um padrão com seu respectivo valor desejado de saída (t_j) e verificar a saída da rede (y_j);
- 3) calcula o erro na saída $E_j = t_j - y_j$;
- 4) se $E_j = 0$, volta ao passo 2; se $E_j \neq 0$, atualiza os pesos: $\Delta w_{ij} = \eta x_i E_j$;
- 5) volta ao passo 2.

RAFR REDES NEURAIS - RAFR 20

LABIC **ALGORITMO DE APRENDIZADO**

IMPORTANTE

- não ocorre variação no peso se a saída estiver correta;
- caso contrário, cada peso é incrementado de η quando a saída é menor que o valor-alvo e decrementado de η quando a saída é maior que o valor-alvo.

$$\Delta w_{ij} = \eta x_i e_j$$

RAFR REDES NEURAIS - RAFR 21

LABIC **Gradiente de uma função**

Gradiente: $\nabla f(x, y) = \left(\frac{\partial}{\partial x} f(x, y), \frac{\partial}{\partial y} f(x, y) \right)$

Derivada direcional: $D_u f(x, y) = \nabla f(x, y) \cdot \mathbf{u}$

$= \|\nabla f(x, y)\| \|\mathbf{u}\| \cos \gamma$

$= \|\nabla f(x, y)\| \cos \gamma$

$D_u f(x, y)$ é a taxa de variação de $f(x, y)$ na direção definida por \mathbf{u} .

RAFR REDES NEURAIS - RAFR 22

LABIC **Gradiente de uma função**

Teorema do gradiente: Seja f uma função de duas variáveis, diferenciáveis no ponto $P(x, y)$.

- i) O máximo de $D_u f(x, y)$ em $P(x, y)$ é $\|\nabla f(x, y)\|$.
- ii) O máximo da taxa de crescimento de $f(x, y)$ em $P(x, y)$ ocorre na direção de $\nabla f(x, y)$.

Corolário: Seja f uma função de duas variáveis, diferenciáveis no ponto $P(x, y)$.

- i) O mínimo de $D_u f(x, y)$ em $P(x, y)$ é $-\|\nabla f(x, y)\|$.
- ii) O máximo da taxa de **decréscimo** de $f(x, y)$ em $P(x, y)$ ocorre na direção de $-\nabla f(x, y)$.

RAFR REDES NEURAIS - RAFR 23

LABIC **Superfície de Erro** **Processo de Minimização**

A direção do gradiente negativo é a de descida mais íngreme ("steepest descent")

RAFR REDES NEURAIS - RAFR 24

LABIC

Método do Gradiente Descendente (GD)

$$\Delta w_{ij} = -\eta \frac{\delta E_j}{\delta w_{ij}}$$

Cada *peso sináptico* w_{ij} do elemento processador j é atualizado proporcionalmente ao *negativo da derivada parcial do erro* deste processador com relação ao peso.

REDES NEURAIS - RAFR 25

LABIC

Logo:

$$\Delta w_{ij} = -\eta \frac{\delta E_j}{\delta w_{ij}} = -\eta \frac{\delta E_j}{\delta s_j} \frac{\delta s_j}{\delta w_{ij}}$$

$$E_j = \frac{1}{2} (t_j - s_j)^2$$

$$s_j = k \Sigma x_i \cdot w_{ij} + \theta$$

$$\Delta w_{ij} = -\eta \cdot [2 \cdot \frac{1}{2} (t_j - s_j) \cdot (-1)] \cdot x_i$$

$$= -\eta \cdot [-(t_j - s_j)] \cdot x_i = \eta x_i (t_j - s_j)$$

REDES NEURAIS - RAFR 26

LABIC

EXEMPLO

Simulação do Operador Lógico AND

AND	x_0	x_1	x_2	t
Entrada 1:	1	0	0	0
Entrada 2:	1	0	1	0
Entrada 3:	1	1	0	0
Entrada 4:	1	1	1	1

Peso inicial: $w_0 = 0, w_1 = 0, w_2 = 0$
Taxa de aprendizado: $\eta = 0.5$

Estrutura da Rede

REDES NEURAIS - RAFR 27

LABIC

EXEMPLO

1º Ciclo

Entrada 1: $s_{out} = f(w_0 x_0 + w_1 x_1 + w_2 x_2)$
 $= f(0 \times 1 + 0 \times 0 + 0 \times 0) = f(0) = 0 \rightarrow s_{out} = t$

Entrada 2: $s_{out} = f(w_0 x_0 + w_1 x_1 + w_2 x_2)$
 $= f(0 \times 1 + 0 \times 1 + 0 \times 0) = f(0) = 0 \rightarrow s_{out} = t$

Entrada 3: $s_{out} = f(w_0 x_0 + w_1 x_1 + w_2 x_2)$
 $= f(0 \times 1 + 0 \times 0 + 0 \times 1) = f(0) = 0 \rightarrow s_{out} = t$

Entrada 4: $s_{out} = f(w_0 x_0 + w_1 x_1 + w_2 x_2)$
 $= f(0 \times 1 + 0 \times 1 + 0 \times 1) = f(0) = 0 \rightarrow s_{out} \neq t$

$w_0 = w_0 + (t - s_{out}) x_0 = 0 + 0.5 \times (1 - 0) \times 1 = 0.5$
 $w_1 = w_1 + (t - s_{out}) x_1 = 0 + 0.5 \times (1 - 0) \times 1 = 0.5$
 $w_2 = w_2 + (t - s_{out}) x_2 = 0 + 0.5 \times (1 - 0) \times 1 = 0.5$

REDES NEURAIS - RAFR 28

LABIC

EXEMPLO

2º Ciclo

Entrada 1: $s_{out} = f(w_0 x_0 + w_1 x_1 + w_2 x_2)$
 $= f(0.5 \times 1 + 0.5 \times 0 + 0.5 \times 0) = f(0.5) = 1 \rightarrow s_{out} \neq t$

$w_0 = w_0 + (t - s_{out}) x_0 = 0.5 + 0.5 \times (0 - 1) \times 1 = 0$
 $w_1 = w_1 + (t - s_{out}) x_1 = 0.5 + 0.5 \times (0 - 1) \times 0 = 0.5$
 $w_2 = w_2 + (t - s_{out}) x_2 = 0.5 + 0.5 \times (0 - 1) \times 0 = 0.5$

Entrada 2: $s_{out} = f(w_0 x_0 + w_1 x_1 + w_2 x_2)$
 $= f(0 \times 1 + 0.5 \times 0 + 0.5 \times 1) = f(0.5) = 1 \rightarrow s_{out} \neq t$

$w_0 = w_0 + (t - s_{out}) x_0 = 0 + 0.5 \times (0 - 1) \times 1 = -0.5$
 $w_1 = w_1 + (t - s_{out}) x_1 = 0.5 + 0.5 \times (0 - 1) \times 0 = 0.5$
 $w_2 = w_2 + (t - s_{out}) x_2 = 0.5 + 0.5 \times (0 - 1) \times 1 = 0$

REDES NEURAIS - RAFR 29

LABIC

EXEMPLO

2º Ciclo

Entrada 3: $s_{out} = f(w_0 x_0 + w_1 x_1 + w_2 x_2)$
 $= f(-0.5 \times 1 + 0.5 \times 1 + 0 \times 0) = f(0) = 0 \rightarrow s_{out} = t$

Entrada 4: $s_{out} = f(w_0 x_0 + w_1 x_1 + w_2 x_2)$
 $= f(-0.5 \times 1 + 0.5 \times 1 + 0 \times 1) = f(0) = 0 \rightarrow s_{out} \neq t$

$w_0 = w_0 + (t - s_{out}) x_0 = -0.5 + 0.5 \times (1 - 0) \times 1 = 0$
 $w_1 = w_1 + (t - s_{out}) x_1 = 0.5 + 0.5 \times (1 - 0) \times 1 = 1$
 $w_2 = w_2 + (t - s_{out}) x_2 = 0 + 0.5 \times (1 - 0) \times 1 = 0.5$

REDES NEURAIS - RAFR 30

EXEMPLO

LABIC

3º Ciclo

Entrada 1: $s_{out} = f(w_0x_0 + w_1x_1 + w_2x_2)$
 $= f(0 \times 1 + 1 \times 0 + 0.5 \times 0) = f(0) = 0 \rightarrow s_{out} = t$

Entrada 2: $s_{out} = f(w_0x_0 + w_1x_1 + w_2x_2)$
 $= f(0 \times 1 + 1 \times 0 + 0.5 \times 1) = f(0.5) = 1 \rightarrow s_{out} \neq t$

$w_0 = w_0 + (t - s_{out})x_0 = -0.5 + 0.5 \times (0 - 1) \times 1 = -1$
 $w_1 = w_1 + (t - s_{out})x_1 = 1 + 0.5 \times (0 - 1) \times 0 = 1$
 $w_2 = w_2 + (t - s_{out})x_2 = 0.5 + 0.5 \times (0 - 1) \times 1 = 0$

REDES NEURAIS - RAFR 31

EXEMPLO

LABIC

3º Ciclo

Entrada 3: $s_{out} = f(w_0x_0 + w_1x_1 + w_2x_2)$
 $= f(-1 \times 1 + 1 \times 1 + 0 \times 0) = f(0) = 0 \rightarrow s_{out} = t$

Entrada 4: $s_{out} = f(w_0x_0 + w_1x_1 + w_2x_2)$
 $= f(-1 \times 1 + 1 \times 1 + 0 \times 1) = f(0) = 0 \rightarrow s_{out} \neq t$

$w_0 = w_0 + (t - s_{out})x_0 = -1 + 0.5 \times (1 - 0) \times 1 = -0.5$
 $w_1 = w_1 + (t - s_{out})x_1 = 1 + 0.5 \times (1 - 0) \times 1 = 1.5$
 $w_2 = w_2 + (t - s_{out})x_2 = 0 + 0.5 \times (1 - 0) \times 1 = 0.5$

REDES NEURAIS - RAFR 32

EXEMPLO

LABIC

4º Ciclo

Entrada 1: $s_{out} = f(w_0x_0 + w_1x_1 + w_2x_2)$
 $= f(-0.5 \times 1 + 1.5 \times 0 + 0.5 \times 0) = f(-0.5) = 0 \rightarrow s_{out} = t$

Entrada 2: $s_{out} = f(w_0x_0 + w_1x_1 + w_2x_2)$
 $= f(-0.5 \times 1 + 1.5 \times 0 + 0.5 \times 1) = f(0) = 0 \rightarrow s_{out} = t$

Entrada 3: $s_{out} = f(w_0x_0 + w_1x_1 + w_2x_2)$
 $= f(-0.5 \times 1 + 1.5 \times 1 + 0.5 \times 0) = f(1) = 1 \rightarrow s_{out} \neq t$

$w_0 = w_0 + (t - s_{out})x_0 = -0.5 + 0.5 \times (0 - 1) \times 1 = -1$
 $w_1 = w_1 + (t - s_{out})x_1 = 1.5 + 0.5 \times (0 - 1) \times 1 = 1$
 $w_2 = w_2 + (t - s_{out})x_2 = 0.5 + 0.5 \times (0 - 1) \times 0 = 0.5$

Entrada 4: $s_{out} = f(w_0x_0 + w_1x_1 + w_2x_2)$
 $= f(-1 \times 1 + 1 \times 1 + 0.5 \times 1) = f(0.5) = 1 \rightarrow s_{out} = t$

REDES NEURAIS - RAFR 33

EXEMPLO

LABIC

5º Ciclo

Entrada 1: $s_{out} = f(w_0x_0 + w_1x_1 + w_2x_2)$
 $= f(-1 \times 1 + 1 \times 0 + 0.5 \times 0) = f(-1) = 0 \rightarrow s_{out} = t$

Entrada 2: $s_{out} = f(w_0x_0 + w_1x_1 + w_2x_2)$
 $= f(-1 \times 1 + 1 \times 0 + 0.5 \times 1) = f(-0.5) = 0 \rightarrow s_{out} = t$

Entrada 3: $s_{out} = f(w_0x_0 + w_1x_1 + w_2x_2)$
 $= f(-1 \times 1 + 1 \times 1 + 0.5 \times 0) = f(0) = 0 \rightarrow s_{out} = t$

Entrada 4: $s_{out} = f(w_0x_0 + w_1x_1 + w_2x_2)$
 $= f(-1 \times 1 + 1 \times 1 + 0.5 \times 1) = f(0.5) = 1 \rightarrow s_{out} = t$

$w_0 = -1, w_1 = 1, w_2 = 0.5$

REDES NEURAIS - RAFR 34

INTERPRETAÇÃO GEOMÉTRICA

LABIC

Linha de Decisão:
 $x_1w_1 + x_2w_2 = -\theta$
 \downarrow
 $x_1 + 0.5x_2 = 1$

REDES NEURAIS - RAFR 35

Perceptron

LABIC

■ **OBS:** A classe de funções representadas por Perceptrons é Limitada

2) Perceptrons Não-Lineares

$Out(\underline{x}) = g(\underline{w}^t \underline{x}) \rightarrow y$

$Out(\underline{x}) = g(\underline{w}^T \underline{x}) = g(\sum_i w_i x_i)$ onde g é uma função não-linear (SIGMOID)

REDES NEURAIS - RAFR 36

O PROBLEMA DO OU-EXCLUSIVO (XOR)

LABIC

PONTO	x_1	x_2	Saída
A_0	0	0	0
A_1	0	1	1
A_2	1	0	1
A_3	1	1	0

Função Degrau

De acordo com a definição do neurônio: $s = F(x_1 w_1 + x_2 w_2 + \theta)$

$net = x_1 w_1 + x_2 w_2 + \theta$

Se $net \geq 0 \rightarrow s = 1$
Se $net < 0 \rightarrow s = 0$

A rede Perceptron divide o plano x_1, x_2 em duas regiões (através da reta net)

RAFR - RAFR 37

O PROBLEMA DO OU-EXCLUSIVO (XOR)

LABIC

Função AND

Função OR

Função OU-Exclusivo

REDES NEURAIS - RAFR 38

O PROBLEMA DO OU-EXCLUSIVO (XOR)

LABIC

Conclusão

- mudando-se os valores de w_1, w_2 e θ , muda-se a inclinação e a posição da reta;
- entretanto é impossível achar uma reta que divide o plano de forma separar os pontos A_1 e A_2 de um lado e A_0 e A_3 de outro
- redes de 1 única camada só representam **funções linearmente separáveis**

RAFR - RAFR 39

O PROBLEMA DO OU-EXCLUSIVO (XOR)

LABIC

Minsky & Papert provaram que este problema pode ser solucionado adicionando-se uma outra camada intermediária de processadores- Multi-Layer Perceptron (MLP)

REDES NEURAIS - RAFR 40

O PROBLEMA DO OU-EXCLUSIVO (XOR)

LABIC

Exemplo:

$w_{11} = +1, w_{12} = -1, w_{21} = -1, w_{22} = +1, \theta_1 = -0.5, \theta_2 = +1$

$w_{11} = w_{12} = w_{21} = w_{22} = +1$

$s_1 = 1 \rightarrow s_1 w_{11} + s_2 w_{21} + \theta_1 \geq 0$
 $-2s_1 + s_2 - 0.5 \geq 0$
 $-2s_1 + s_2 \geq 0.5$

s_1 é inibitório e s_2 é excitatório

RAFR - RAFR 41

O PROBLEMA DO OU-EXCLUSIVO (XOR)

LABIC

Exemplo do OU-EXCLUSIVO:

- $J = 2$ (número de entradas originais - x_1, x_2)
- $H = 1$ (número de entradas adicionais - $x_1 \cdot x_2$)

Pontos	Entradas	Saída
A	-1 -1 1	-1
B	-1 1 -1	1
C	1 -1 -1	1
D	1 1 -1	-1

Problema Linearmente Separável

REDES NEURAIS - RAFR 42

LABIC

MULTI-LAYER PERCEPTRON

- Redes de apenas uma camada só representam funções linearmente separáveis
- Redes de múltiplas camadas solucionam essa restrição
- O desenvolvimento do algoritmo Back-Propagation foi um dos motivos para o ressurgimento da área de redes neurais

REDES NEURAIS - RAFR 43

LABIC

Algoritmos de Aprendizado

- **Problema**

\underline{x}_1	y_1
\underline{x}_2	y_2
\underline{x}_3	y_3
\underline{x}_N	y_N

\underline{x} denota um vetor de p componentes

REDES NEURAIS - RAFR 44

LABIC

- Perceptrons Lineares: eles são modelos multivariados lineares:

$$\text{Out}(\underline{x}) = \underline{w}^T \underline{x} \rightarrow y$$
- e o treinamento consiste de minimizar a soma dos quadrados do resíduo pelo método do “gradiente descent”.
- Regra Gradiente Descent:

$$w \leftarrow w - \eta \partial f(w) / \partial w$$

REDES NEURAIS - RAFR 45

LABIC

Adaline

- O objetivo do processo adaptativo do Adaline consiste em utilizar a função de ativação “hard limiter (saída +1 ou -1)” e minimizar os pesos e θ usando o algoritmo LMS.
- Como a saída desejada pode ter apenas valores 1 ou -1, o erro pode ser +2, 0, ou -2.

REDES NEURAIS - RAFR 46

LABIC

Regra Delta ou LMS

- Estamos interessados em encontrar uma regra de aprendizado que permite encontrar o vetor \underline{w} por melhoramentos sucessivos.
- Função custo: $E(\underline{w}) = 0.5 \sum_i (y_i - \underline{w} x_i)^2$
- caminhando em direção oposta ao gradiente, usando o alg. “Gradiente descent”, podemos encontrar \underline{w} .

REDES NEURAIS - RAFR 47

LABIC

Regrad Delta ou LMS

$$\Delta w_{jk} = - \frac{\partial E}{\partial w_{jk}}$$

$$\Delta w_{jk} = -\eta (out_j^p - y_j^p) x_k^p$$

- que correspondem as mudanças no padrão p

Definindo: $\delta_j = (out_j^p - y_j^p)$

REDES NEURAIS - RAFR 48



Adaline

$$\Delta w_{jk} = -\eta \delta_j^p x_k^p$$

REGRA **DELTA**, REGRA **LMS** ou
REGRA de **Widrow-Hoff (1960)**



Exercícios

- Pensar em uma rede de perceptrons constituída de uma camada intermediária constituída de um único neurônio, para implementar a função OU-EXCLUSIVO.
- Implementar uma ADALINE para reconhecer a letra A e a letra A invertida.